M&G No. 60001.0287US01/305237.01

REPRESENTING USER EDIT PERMISSION OF REGIONS WITHIN AN ELECTRONIC DOCUMENT

Related Applications

This patent application is a continuation-in-part application under 35 United States Code § 120 of United States Patent Application No. 10/187,060 filed on June 28, 2002, which is incorporated herein by reference. An exemplary schema in accordance with the present invention is disclosed beginning on page 11 in an application entitled "Mixed Content Flexibility," Serial No. _____, Docket No. 60001.0275US01, filed December 2, 2003, which is hereby incorporated by reference in its entirety.

Background of the Invention

Markup Languages have attained wide popularity in recent years. One type of markup language, Extensible Markup Language (XML), is a universal language that provides a way to identify, exchange, and process various kinds of data. For example, XML is used to create documents that can be utilized by a variety of application programs. Elements of an XML file have an associated namespace and schema.

In XML, a namespace is a unique identifier for a collection of names that are used in XML documents as element types and attribute names. The name of a namespace is commonly used to uniquely identify each class of XML document. The unique namespaces differentiate markup elements that come from different sources and happen to have the same name.

XML Schemata provide a way to describe and validate data in an XML environment. A schema states what elements and attributes are used to describe content in an XML document, where each element is allowed, what types of text contents are allowed within it and which elements can appear within which other elements. The use

of schemata ensures that the document is structured in a consistent manner. Schemata may be created by a user and generally supported by an associated markup language, such as XML. By using an XML editor, the user can manipulate the XML file and generate XML documents that adhere to the schema the user has created. XML documents may be created to adhere to one or more schemata.

Electronic documents are often edited by multiple users. For example, a document can be written by a reporter, and then modified by a news editor. In other cases, a former document can be provided by an employer, which is then "filled out" by employees. In such cases, it is desirable to only allow certain regions of the electronic document to be edited. In some cases, electronic documents may be edited by applications that do not "understand" the file protection scheme used by the application that was used to originally author the electronic documents. What is needed is a way to easily handle a variety of protection settings and editing rights when transforming a document into an ML format.

Summary of the Invention

The present invention is directed towards defining regions within editable objects of electronic document such that specific editing permissions can be granted to specific users for specific regions. The regions can be expressed in an ML format such that a variety of applications that consume ML content can operate in accordance with the granted permissions.

According to one aspect of the invention, a computer-readable medium having computer-executable components comprises four components. The first component is arranged to edit an electronic document having editable objects. An editable object can be, for example, a selectable region that comprises paragraphs, characters, tables, images, rows, cells, columns, and/or text. The second component is arranged to define a first location for the start of an editable object region for which a level of editing permission for a specific user is desired and to define a second location for the end of the editable object region. The third component is arranged to associate a user identifier for the specific user with the text region that is defined by the first and second locations. The fourth component is arranged to encode in an ML format the

electronic document, a first element that defines the first location, and a second element that defines the second location, wherein one of the first and second element further comprises the user identifier.

According to another aspect of the invention, a method for handling electronic documents comprises editing an electronic document having editable objects. A first location is defined for the start of an editable object region for which a level of editing permission for a specific user is desired. A second location is defined for the end of the editable object region. A user identifier for the specific user is associated with the text region that is defined by the first and second locations. The electronic document, a first element that defines the first location, and a second element that defines the second location are encoded in an ML format, wherein one of the first and second elements further comprises the user identifier.

According to yet another aspect of the invention, a system for displaying and modifying electronic documents comprises an electronic document file, an editor, and an encoder. The electronic document file comprises editable objects. The editor is arranged to define a first location for the start of an editable object region for which a level of editing permission for a specific user is desired, to define a second location for the end of the editable object region, and to associate a user identifier for the specific user with the text region that is defined by the first and second locations. The encoder is configured to encode in an ML format the electronic document, a first element that defines the first location, and a second element that defines the second location, wherein one of the first and second elements further comprises the user identifier.

Brief Description of the Drawings

FIGURE 1 illustrates an exemplary computing device that may be used in one exemplary embodiment of the present invention.

FIGURE 2 is a block diagram illustrating an exemplary environment for practicing the present invention.

FIGURE 3 illustrates an exemplary ML file in accordance with aspects of the present invention.

FIGURE 4 illustrates an exemplary protection element, in accordance with aspects of the present invention.

FIGURE 5 illustrates an exemplary permission start element and permission end element, in accordance with aspects of the present invention.

FIGURE 6 illustrates of a process flow for representing user edit permission regions within an electronic document, in accordance with aspects of the invention.

Detailed Description of the Preferred Embodiment

Throughout the specification and claims, the following terms take the meanings explicitly associated herein, unless the context clearly dictates otherwise.

The terms "markup language" or "ML" refer to a language for special codes within a document that specify how parts of the document are to be interpreted by an application. In a word-processor file, the markup language specifies how the text is to be formatted or laid out, whereas in a particular customer schema, the ML tends to specify the text's meaning according to that customer's wishes (e.g., customerName, address, etc). The ML is typically supported by a word-processor and may adhere to the rules of other markup languages, such as XML, while creating further rules of its own.

The term "element" refers to the basic unit of an ML document. The element may contain attributes, other elements, text, and other building blocks for an ML document.

The term "tag" refers to a command inserted in a document that delineates elements within an ML document. Each element can have no more than two tags: the start tag and the end tag. It is possible to have an empty element (with no content) in which case one tag is allowed.

The content between the tags is considered the element's "children" (or descendants). Hence other elements embedded in the element's content are called "child elements" or "child nodes" or the element. Text embedded directly in the content of the element is considered the element's "child text nodes". Together, the child elements and the text within an element constitute that element's "content".

The term "attribute" refers to an additional property set to a particular value and associated with the element. Elements may have an arbitrary number of attribute settings associated with them, including none. Attributes are used to associate additional information with an element that will not contain additional elements, or be treated as a text node.

Illustrative Operating Environment

With reference to FIGURE 1, one exemplary system for implementing the invention includes a computing device, such as computing device 100. In a very basic configuration, computing device 100 typically includes at least one processing unit 102 and system memory 104. Depending on the exact configuration and type of computing device, system memory 104 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory 104 typically includes an operating system 105, one or more applications 106, and may include program data 107. In one embodiment, application 106 may include a word-processor application 120 that further includes ML editor 122. This basic configuration is illustrated in FIGURE 1 by those components within dashed line 108.

Computing device 100 may have additional features or functionality. For example, computing device 100 may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in FIGURE 1 by removable storage 109 and non-removable storage 110. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory 104, removable storage 109 and non-removable storage 110 are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired

information and which can be accessed by computing device 100. Any such computer storage media may be part of device 100. Computing device 100 may also have input device(s) 112 such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) 114 such as a display, speakers, printer, etc. may also be included. These devices are well know in the art and need not be discussed at length here.

Computing device 100 may also contain communication connections 116 that allow the device to communicate with other computing devices 118, such as over a network. Communication connection 116 is one example of communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media.

Word-Processor File Structure

FIGURE 2 is a block diagram illustrating an exemplary environment for practicing the present invention. The exemplary environment shown in FIGURE 2 is a word-processor environment 200 that includes word-processor 120, ML file 210, ML Schema 215, and ML validation engine 225.

In one embodiment, word-processor 120 has its own namespace or namespaces and a schema, or a set of schemas, that is defined for use with documents associated with word-processor 120. The set of tags and attributes defined by the schema for word-processor 120 define the format of a document to such an extent that it is referred to as its own native ML.

Word-processor 120 internally validates ML file 210. When validated, the ML elements are examined as to whether they conform to the ML schema 215. As

previously described above, a schema states what tags and attributes are used to describe content in an ML document, where each tag is allowed, and which tags can appear within other tags, ensuring that the documentation is structured the same way. Accordingly, ML 210 is valid when structured as set forth in arbitrary ML schema 215.

ML validation engine 225 operates similarly to other available validation engines for ML documents. ML validation engine 225 evaluates ML that is in the format of the ML validation engine 225. For example, XML elements are forwarded to an XML validation engine. In one embodiment, a greater number of validation engines may be associated with word-processor 120 for validating a greater number of ML formats.

FIGURE 3 illustrates an exemplary ML file in accordance with aspects of the present invention. ML file 300 includes ML elements. An element in a markup language usually includes an opening tag (indicated by a "<" and ">"), some content, and a closing tag (indicated by a "</" and ">"). In this example, tags associated with ML include a "w:" within the tag (e.g., 302). The "w:" prefix is used as shorthand notation for the namespace associated with the element.

The text contained within the document follows the "T" tag, making it relatively easy for an application to extract the text content from a word-processing document created in accordance with aspects of the invention. Given that the example shown is valid, ML file 210 produces a document with a body and two paragraphs that include the text "Work" in the first paragraph and "123 Main" in the second paragraph.

The text contained within the document can be displayed according to styles that can be declared in the ML file. Typically, the styles declarations are declared near the top of the ML file, which allows the styles to be referenced by various objects in the body of the document. For example, a root element "w:wordDocument" can be used to declare the child element "w:styles," which can be used to store the style definitions.

FIGURE 4 illustrates an exemplary protection element, in accordance with aspects of the present invention. The figure illustrates listing 400 having a protection element (410) in which the example attributes are listed. Protection element

410 comprises attributes including an edit attribute (420), an enforcement attribute (430), and a password protection attribute (440).

Protection element 410 ("document Protection") is typically a child element of a "docPr" element. The (parent) docPr element is typically used to represent document-wide settings, which can apply to elements that are associated with electronic documents as well as the entire document itself.

Protection element 410 is used to represent an electronic document's overall protection settings or state. The protection settings may include settings such as a protection mode, a password (if any), or whether protection enforcement has been activated (or not). The protection settings can be included as attributes of protection element 410.

Editing attributes can be used to impose certain editing the restrictions on an entire document (or sections). For example, a user can specify that an electronic document can only be edited with the "track changes" option enabled. Additionally, the user can specify that while only comments can be inserted into the document, the actual contents of the document cannot be changed. Furthermore, the author of the document can specify that nothing in the document can be changed and that the document is restricted to " read-only" access only.

It is also possible to override the user settings that specify how the document or arbitrary pieces of text within a document can be protected. For example, specific users (or users that are associated with specific workgroups) can override the restrictions a user places on an electronic document such that named users (or workgroups) can have fuller editing rights to those pieces (or regions) of text. These specific users can be given full editing rights, or restricted editing rights that are restricted differently from the document-wide protection mode.

Accordingly, the editing capabilities of a user who has not been given special editing rights to a given region of text is typically limited to just the editing capabilities that are determined by the document-level protection. Alternatively, those users who have been assigned editing rights that are specific to regions of text can typically edit the region of text more freely than what the level of protection applied to

the entire document / section will allow. This give those users / groups that have been granted permission more flexibility than the users who have not been assigned special editing rights.

Edit attribute 420 can be used to represent those types of protection that can be applied to unauthorized or unauthenticated users at the document or section level. Possible values for edit attribute 420 include "read-only," which can be used to prevent the document from being edited in any way. Another possible value is "comments," which can be used to allow only comments to be inserted into the document. Other values for various editing restrictions can be named such as, for example the "track changes" protection mode described above.

Enforcement attribute 430 can be used to indicate whether the protection mode is actually enabled. Possible values for the enforcement attribute include "on" for when protection enforcement is enabled, and "off" for when protection enforcement is not enabled. Other possible values may be named for intermediate levels of enforcement, to enable degrees of enforcement between being fully enabled and completely disabled modes of enforcement.

Password protection attribute 440 can be used to indicate whether a password is to be used to protect a defined region of text. (A defined region of text may include only a portion of the text within an electronic document or all of the text within electronic document.) The value for the password protection attribute can be an encrypted version of the password itself (or, for example, a digital signature of the password). Alternatively, possible values for the password attribute may just be blank, in the case of there being no password specified.

Accordingly, Figure 4 illustrates an example instance of protection element 410. The example instance of protection element 410 comprises edit attribute 420 having a value of "read-only," enforcement attribute 430 having a value of "on," and password protection attribute 440 having a value that is the encrypted password.

FIGURE 5 illustrates an exemplary permission start element and permission end element, in accordance with aspects of the present invention. The figure illustrates an example listing 500, which provides a demonstration of the usage of

permission start elements 510 and 540. Permission start element 510 is associated with permission end element 570 and permission start element 540 is associated with permission and element 560.

The permission start element and the associated permission end element define a region of text that can be used to indicate the beginning of a region of text for which the document's editing restrictions are overridden for specified users and/or workgroups. The permission start element is a possible sibling of a paragraph element (e.g., 525) or a text run element (e.g., 530), which is used to represent the text run inside a paragraph, as well as any other object that the editing permissions may be applied to. In one instance, a separate permission start element may be used for each user or workgroup for which protection is desired. It may also be possible to specify all the users in one permission tag if that group's permissions are the same.

The permission start element (e.g., 510) typically comprises attributes including an identifier attribute (515); and one or more of an edit attribute (520) and/or an edit group identifier (515). The identifier attribute is used to uniquely identify the pairing of the permission start tag with the permission end tag. The identified user is granted specific permission to edit the specific region of text according to the specific permission that is associated with the identifier attribute by the application that is used to edit the file.

The editor element is typically a string that identifies the individual user to which this permission is assigned. The editor element can be, for example, an operating system user name, an e-mail address, or any other string that can be used to uniquely identify the user to which the permission belongs.

The group better element is typically a string that identifies a named group to which this permission is assigned. The group editor element can be, for example, it could be a built-in (with respect to the operating system) group named such as "everyone", other aliases that are used to represent groups of one or more individuals, and the like.

A permission end element (e.g., 560) is used to indicate the end of a region of text having editing permissions that are associated with a corresponding

permission start element (e.g., 540). The permission end element comprises an identifier attribute that provides a one-to-one correspondence with the identifier attribute of a corresponding permission start element. This enables for the corresponding start and end element to be properly identified and paired. This approach is preferable to using just one tag to span the entire permission range, because it ensures that a well formed XML document is produced. (Well formed XML documents are disclosed in United States Patent Application Serial No. _____, filed on December 3, 2003, Docket No. 60001.0283US01, entitled "REPRESENTING NON-STRUCTURED FEATURES IN A WELL FORMED DOCUMENT," which is hereby incorporated by reference.)

The permission end element is a possible sibling of a paragraph element or a text run element. Accordingly, each permission end element is associated with a corresponding permission start element. Each permission end element is matched with the corresponding permission start element by matching the identifier attribute of the permission end element with the identifier attribute of the permission start element. The matching identifier attributes allow the permission start element and the permission end element to be associated having a one-to-one correspondence.

The example markup language in the figure represents the sentence "the quick brown fox jumps over the lazy dog." The representative sentence as a whole is editable by an individual identified as ms@domainname.com. Additionally, the word "fox" is editable by a group identified as "everyone."

Although the text region identified by permission start element 540 is shown as being nested within the text region identified by permission start element 510, it is possible to overlap the text regions. For example, exchanging the identifier attributes of permission end elements 560 and 570 would result in overlapping text regions.

FIGURE 6 illustrates of a process 600 flow for representing user edit permission regions within an electronic document, in accordance with aspects of the invention. After a start block, the process moves to block 610, at which point a document is opened for editing by a user who would like to restrict user edit permissions for text regions within the opened document. The selection of the file for

opening may, for example, include highlighting the selected file within a file browser. The document may be, for example, a document that includes paragraphs and text runs. The document may also be, for example, a document that includes spreadsheet cells or word-processor paragraphs. The document can be stored in a proprietary format of the editor process.

At block 615, the location for the beginning of a user edit permission region (i.e., the permission start element) is selected. The selection of the beginning of the region may, for example, include using a mouse to pilot a cursor to a specific location within the document and depressing a button on the mouse.

Continuing at block 620, the location for the end of a user edit permission region (i.e., the permission end element) is selected. The selection of the end of the region may, for example, include dragging mouse to a second location within the document and releasing the depressed mouse button.

At block 625, a value is selected for the editor (or group editor) attribute of the permission start element. The value can, for example, default to "everyone" or can be specified by the user who can supply a user name within a dialog box. A unique identifier is typically assigned to the permission start element and the permission end element such that the same identifier is assigned to both. Additionally a level of editing permissions can be selected by the user, which can be encoded within the unique identifier that is assigned to both the permission start and permission end elements.

At block 630, the document is saved using an ML format. The beginning and the ending locations for the selected user edit permission region are encoded in the saved file using a permission start element and a permission end element. The permission start element contains a unique identifier that typically matches the unique identifier of the permission end element. The permission start element also typically contains the editor attribute (which specifies the user or group that has permission to edit the region). Alternatively (or conjunctively), the permission end element could be defined to store the editor attribute.

As a further illustration of representing user edit permission regions within an electronic document within electronic documents, a schema is provided as follows:

Document Properties for Protection State:

```
<xsd:element name="documentProtection" type="docProtectProperty"
minOccurs="0">
      <xsd:annotation>
            <xsd:documentation>Helps prevent unintentional changes to
all or part of an online form or document, as specified (Protect
Document option).</xsd:documentation>
      </xsd:annotation>
</xsd:element>

<xsd:complexType name="docProtectProperty">
      <xsd:annotation>
            <xsd:documentation>Helps prevent unintentional changes to
all or part of an online form or document as
specified.</xsd:documentation>
      </xsd:annotation>
      <xsd:attribute name="edit" type="docProtectValue"
use="optional">
            <xsd:annotation>
                  <xsd:documentation>Gets or sets editing
restrictions. Helps prevent unintentional editing changes as
specified.</xsd:documentation>
            </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="formatting" type="onOffType"
use="optional">
            <xsd:annotation>
                  <xsd:documentation>Gets or sets formatting
restrictions. Prevents unintentional formatting changes except as
allowed.  This setting does not have an effect unless the
formattingEnabled attribute is on.</xsd:documentation>
            </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="enforcement" type="onOffType">
            <xsd:annotation>
                  <xsd:documentation>Gets or sets whether the
specified restrictions are currently being enforced for this
document.</xsd:documentation>
            </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="unprotectPassword" type="longHexNumberType"
use="optional">
            <xsd:annotation>
                  <xsd:documentation>Gets or sets password key to
unprotect this document from unintentional formatting/editing changes.
This password is not secure.</xsd:documentation>
            </xsd:annotation>
      </xsd:attribute>
</xsd:complexType>
```

```
<xsd:simpleType name="docProtectValue">
      <xsd:annotation>
            <xsd:documentation>Defines document-protection editing-
restriction values.</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
            <xsd:enumeration value="none">
                  <xsd:annotation>
                        <xsd:documentation>No document protection;
reviewers may make any changes to the document.</xsd:documentation>
                  </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="read-only">
                  <xsd:annotation>
                        <xsd:documentation>Let's reviewers make no
changes to the document.</xsd:documentation>
                  </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="comments">
                  <xsd:annotation>
                        <xsd:documentation>Let's reviewers insert
comments but does not let reviewers change the contents of the
document.</xsd:documentation>
                  </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="tracked-changes">
                  <xsd:annotation>
                        <xsd:documentation>Let's reviewers change a
document but highlights all changes so that the author can track
changes.  While a document is protected for tracked changes, you
cannot turn orr changes tracking nor can you accept or reject tracked
changes.</xsd:documentation>
                  </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="forms">
                  <xsd:annotation>
                        <xsd:documentation>Protects a document from
changes except in form fields or unprotected sections.  To turn
protection on or off for a section see the 'FormProt' element inside
'sectPr'.</xsd:documentation>
                  </xsd:annotation>
            </xsd:enumeration>
      </xsd:restriction>
</xsd:simpleType>
```

Inline Editing Permissions:

```
<xsd:element name="permStart" minOccurs="0" type="permStartElt">
      <xsd:annotation>
            <xsd:documentation>Represents the range protection
permission start.</xsd:documentation>
      </xsd:annotation>
</xsd:element>
<xsd:element name="permEnd" minOccurs="0" type="permElt">
```

```
      <xsd:annotation>
            <xsd:documentation>Represents the range protection
permission end.</xsd:documentation>
      </xsd:annotation>
</xsd:element>


<xsd:complexType name="permStartElt">
      <xsd:annotation>
            <xsd:documentation>Defines the range protection permission
start.</xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
            <xsd:extension base="permElt">
                  <xsd:attribute name="edGrp" type="edGrpType"
use="optional">
                        <xsd:annotation>
                              <xsd:documentation>Gets or sets the
group with edit permissions</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
                  <xsd:attribute name="ed" type="stringType"
use="optional">
                        <xsd:annotation>
                              <xsd:documentation>Gets or sets the user
with edit permissions</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
                  <xsd:attribute name="col-first"
type="decimalNumberType" use="optional">
                        <xsd:annotation>
                              <xsd:documentation>For table range
permissions, specifies the beginning column for the
permission.</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
                  <xsd:attribute name="col-last"
type="decimalNumberType" use="optional">
                        <xsd:annotation>
                              <xsd:documentation>For table range
permissions, specifies the ending column for the
permission.</xsd:documentation>
                        </xsd:annotation>
                  </xsd:attribute>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>


<xsd:simpleType name="edGrpType">
      <xsd:annotation>
            <xsd:documentation>Defines a group with edit
permissions.</xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
```

```
                <xsd:enumeration value="none"></xsd:enumeration>
                <xsd:enumeration value="everyone"></xsd:enumeration>
                <xsd:enumeration value="administrators"></xsd:enumeration>
                <xsd:enumeration value="contributors"></xsd:enumeration>
                <xsd:enumeration value="editors"></xsd:enumeration>
                <xsd:enumeration value="owners"></xsd:enumeration>
                <xsd:enumeration value="current"></xsd:enumeration>
        </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="permElt">
      <xsd:annotation>
            <xsd:documentation>Defines a permission for the
document.</xsd:documentation>
      </xsd:annotation>
      <xsd:attribute name="id" type="stringType" use="required">
            <xsd:annotation>
                  <xsd:documentation>Gets or sets the ID for this
permission.</xsd:documentation>
            </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="displacedBySDT" type="displacedBySDTValue"
use="optional">
            <xsd:annotation>
                  <xsd:documentation>When bookmarks border Structured
Document Tags (SDTs), this attribute ensures that bookmarks are
inserted into the document next to the SDTs. We use this attribute
because SDTs appear in our XML how they logically appear in the Word
document, but not necessarily in the same location as they are in the
document. When we displace the SDTs, we also displace the bookmarks
next to them so that the intended result is in the XML
file.</xsd:documentation>
            </xsd:annotation>
      </xsd:attribute>
</xsd:complexType>
```

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.